Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A device comprising:

    a memory unit including executable software;

    a plurality of class files stored in the memory unit; and,

    a computing unit connected to the memory unit, the computing unit being able to execute a Java Virtual Machine, the computing unit configured to execute ~~executing the executable~~ software for generating one or more ~~a number of cod~~ files from the plurality of class files by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein the number of ~~cod~~ the generated files is less than the number of the plurality of class files and a given generated ~~cod~~ file ~~includes~~ comprises:

    a constant pool created by combining constant pool entries from two or more of the plurality of class files without duplication of entries;

    a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the plurality of class files ~~without duplication of entries~~; and,

    a fixup table for providing information to the Java Virtual Machine for resolving at least one entry in the given generated ~~cod~~ file at link time.

2. (currently amended) The device of claim 1, wherein ~~for the given cod file,~~ the information in the fixup table ~~includes~~ comprises the location of data needed for resolving a symbolic reference ~~for cross-referencing a method not contained~~ in the given generated ~~cod~~ file.

3. (currently amended) The device of claim 1, wherein there are at least two generated files defined as sibling files in a common sibling group, each of the sibling files comprising ~~the given cod file further includes~~ a sibling list for listing other sibling ~~related cod~~ files in the common sibling group, wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets and references to files that are not part of the common sibling group are indicated using symbolic references ~~to define a sibling group and the fixup table of the given cod file further includes indices to the other related cod files specified in the sibling group~~.

4. (currently amended) The device of claim 1, wherein for the given generated ~~cod~~ file, the byte codes and information structure comprises ~~includes~~ a second hard offset for cross-referencing a method included in the given generated ~~cod~~ file that was previously symbolically referenced.

5. (currently amended) The device of claim 3, wherein ~~for the given cod file, the byte codes and information structure includes a~~ at least one of the hard offsets ~~for cross-referencing a method included in the sibling group that was previously symbolically referenced~~ does not need to be resolved or put into context by the Java Virtual Machine at link time.

6. (cancelled).

7. (currently amended) A method for generating one or more ~~a number of cod~~ files from a plurality of class files having constant pools on a computing unit by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein ~~such that~~ the number of the generated ~~cod~~ files is less than the number of the plurality of class files, and for a given generated file ~~without duplication of entries for reducing storage space, wherein for a given cod file,~~ the method comprises:

    identifying class files with common entries in ~~at least one of~~ the constant pools of the class files ~~and the byte codes and information structure~~;

generating a constant pool for the given generated ~~cod~~ file by combining constant pool entries from the plurality of class files with common entries without duplication;

generating the byte codes and information structure for the given generated ~~cod~~ file by combining byte codes and information structure entries from the plurality of class files ~~with common entries without duplication~~; and,

generating a fixup table for providing information to a Java Virtual Machine for resolving at least one entry in the given cod file at link time.

8. (currently amended) The method of claim 7, wherein the method further comprises ~~includes, for the given cod file,~~ providing location of data needed for resolving a symbolic reference in the ~~fixup table for cross-referencing a method not contained in the cod~~ given generated file as the information in the fixup table.

9. (currently amended) The method of claim 7, wherein the method further comprises ~~includes~~: identifying cross-references between the class files, and defining at least two generated files as sibling files in a common sibling group; wherein each of the sibling files comprises

~~generating~~ a sibling list for listing other ~~related cod~~ sibling files in the common ~~to define a~~ sibling group, indicating cross-references between the sibling files in the common sibling group using hard offsets, and indicating references to files that are not part of the common sibling group using symbolic references~~; and,~~

~~providing indices to the other related cod files specified in the sibling group in the fixup table~~.

10. (currently amended) The method of claim 7, wherein the method further comprises ~~includes~~, for the given generated ~~cod~~ file, providing a second hard offset in the byte codes and information structure for cross-referencing a method now included in the given generated ~~cod~~ file that was previously symbolically referenced.

11. (currently amended) The method of claim 9, wherein the method further comprises ~~includes, for the given cod file,~~ providing at least one of the hard offsets in a manner that does not need to be resolved or put into context by the Java Virtual Machine at link time ~~the byte codes and information structure for cross-referencing a method included in the sibling group that was previously symbolically referenced~~.

12. (cancelled).

13. (currently amended) A computer-readable medium storing computer executable instructions ~~An article storing executable software~~ that when executed by a computing unit generates one or more ~~a number of cod~~ files from a plurality of class files by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein the executable software comprises code for generating a given generated ~~cod~~ file to include:

a constant pool created by combining constant pool entries from two or more of the plurality of class files without duplication of entries;

a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the plurality of class files ~~without duplication of entries~~; and,

a fixup table for providing information to a Java Virtual Machine for resolving at least one component of the given generated ~~cod~~ file at link time.

14. (currently amended) The computer-readable medium ~~article~~ of claim 13, wherein ~~for the given cod file,~~ the information in the fixup table comprises the location of data needed for resolving ~~includes~~ a symbolic reference ~~for cross-referencing a method not contained~~ in the given generated ~~cod~~ file.

15. (currently amended) The computer-readable medium ~~article~~ of claim 13, wherein ~~the given cod file further includes~~ there are at least two generated files defined as sibling files in a common sibling group, each of the sibling files comprising a sibling list for listing other ~~related cod~~ sibling files in the common sibling group, wherein

cross-references between the sibling files in the common sibling group are indicated using hard offsets and references to files that are not part of the common sibling group are indicated using symbolic references ~~to define a sibling group and the fixup table of the given cod file further includes indices to the other related cod files specified in the sibling group~~.

16. (currently amended) The computer-readable medium ~~article~~ of claim 13, wherein for the given generated ~~cod~~ file, the executable software comprises code for generating a second hard offset in the byte codes and information structure for cross-referencing a method included in the given generated ~~cod~~ file that was previously symbolically referenced.

17. (currently amended) The computer-readable medium ~~article~~ of claim 15, wherein for the given generated ~~cod~~ file, at least one of the hard offsets does not need to be resolved or put into context by the Java Virtual Machine at link time ~~the executable software comprises code for generating a hard offset in the byte codes and information structure for cross-referencing a method included in the sibling group that was previously symbolically referenced~~.

18. (cancelled).

19. (new) The device of claim 3, wherein the information in the fixup table of a given sibling file comprises the location of data of a cross-referenced sibling file to place one of the hard offsets that corresponds to the cross-referenced sibling file into context at link time.

20. (new) The method of claim 9, wherein the method further comprises providing the location of data of a cross-referenced sibling file as the information in the fixup table of a given sibling file to place one of the hard offsets that corresponds to the cross-reference sibling file into context at link time.

21. (new) The computer-readable medium of claim 15, wherein the information in the fixup table of a given sibling file comprises the location of data of a cross-referenced sibling file to place one of the hard offsets that corresponds to the cross-referenced sibling file into context at link time.